## REMARKS

In view of the foregoing amendments and following remarks responsive to the non-final Office Action dated October 12, 2007, Applicant respectfully requests favorable reconsideration of this application.

### Statutory Subject Matter Rejection

The Office rejected claims 22-28 and 30-35 under 35 U.S.C. 101 as being directed to non-statutory subject matter. The Office argued that these claims are "directed to nothing more than a collection of data, non-functional descriptive material".

Applicant respectfully traverses. Claims 22-28 and 30-35 are computer readable product claims that recite computer executable instructions that cause a computer to perform specified operations. They are not drawn to a collection of data. Applicant notes that the Office has expressly omitted from this rejection claim 29, which depends from claims 22, 27, and 28, which were rejected under 35 U.S.C. 101. Furthermore, claim 29 does not add an additional instruction per se, but merely describes in more detail the sixth instructions previously recited in rejected claims 27 and 28. Applicant cannot discern any significant distinction between the nature of claim 29 and any of the other computer readable product claims from which it depends that would lead to this particular claim not being rejected while the other claims were rejected under 35 U.S.C. 101.

In any event, claims 22-28 and 30-35 clearly are not directed merely to a collection of data. In fact, they do not appear to be directed in any way, shape,

or form to a collection of data let alone exclusively so. Merely as an example, the first set of instructions in claim 1 "provide a graphical user interface ...", the second set of instructions "enable the user to drag and drop symbols into a workspace and label the symbol ...", and the third set of instructions "enable the user to draw lines between objects in the workspace". Each of these sets of instructions calls for instructions that do something, e.g., generate a GUI, enable drag and drop functionality, and enable drawing of lines, respectively. None of these, let alone all of them, are merely collections of data.

For instance, using the first instructions as an example, perhaps it might be said that a claim reciting the GUI itself would be drawn to merely a collection of data. However, the claim recites software for generating a GUI, which is a completely different (and statutory) claim. As another example, let us consider the second instructions which enable dragging and dropping, inter alia. While dragging and dropping per se is not new, let us assume that Applicant was the first to come up with the idea of dragging and dropping GUI items into windows. Clearly, a claim drawn to computer instructions for dragging and dropping would be patentable subject matter. It is not merely a collection of data. Rather, it is a set of computer instructions that enable a very useful computer interface functionality that eases an operator's use of a computer to perform the types of tasks for which computers are used.

Accordingly, Applicant respectfully requests the Office to withdraw this rejection.

**Prior Art Rejections**

Applicant respectfully thanks the Office for the withdrawal of all previous claim rejections. However, the Office has asserted new claim rejections in this latest Office Action. Particularly, the Office has rejected claims 22, 26, 27, and 35 under 35 U.S.C. §102(b) as being anticipated by Bailey; claims 24, 28, 29, 31, and 32 under 35 U.S.C. §103(a) as being unpatentable over Bailey in view of Milakovich; claims 1-21, 23, and 25 under 35 U.S.C. §103(a) as being unpatentable over Bailey in view of Milakovich and further in view of Visio 2000 Standard Edition User Guide (hereinafter "Visio 2000"); and claims 30, 33, and 34 under 35 U.S.C. §103(a) as being unpatentable over Bailey.

Applicant respectfully traverses.

The present invention is a method and software product for defining, representing, and/or documenting object oriented programming applications, and particularly, those developed to work in a graphical user interface. The invention represents the applications in one or more diagrams termed "object and event diagrams" (OEDs). The technique allows an application architect to define the program basis and the program logic without concern about details of the user interface, such as how actions will be executed (which can be determined by the programmer).

Bailey discloses a program development environment for use in generating application programs, but it does not meet all of the limitations of the claims.

For instance, turning first to the rejection of claims 22, 26, 27, and 35 as anticipated by Bailey, Bailey does not meet all of the limitations of independent claim 22. Particularly, claim 22 recites computer executable instructions that enable the user to drag and drop symbols into a work space and label the symbols with a label descriptive of the object's features. The Office asserts that the drag and drop feature is disclosed in column 12, lines 35-37 and that the labeling feature is disclosed in column 9, lines 50-52. However, neither of these assertions are accurate. As can be seen from column 12, lines 33-37, where it states "when a developer selects a particular icon, the program-development environment 310 can direct the window manager 306 to draw the corresponding image from the appropriate bit map. Symbolic representations can also be moved about the designer window 406 by dragging them around with the mouse 230". Thus, when a user selects a symbolic representation of a program feature, he does not drag and drop it into the designer window. Rather, the user clicks on it and it is drawn in the designer window 406. Only after the symbolic representation is in the designer window 406 can the user drag it around.

Accordingly, Bailey does not meet this claim element.

Furthermore, column 9, lines 50-52 disclose that the software of Bailey automatically assigns a name to the symbolic representation, contrary to the language of claim 1 which recites that the instructions "enable the user to...label the symbol with a label descriptive of the object's features".

Accordingly, claim 22 further distinguishes over Bailey based on this claim language.

Dependent claim 27 even further distinguishes over Bailey. Particularly, it depends from claim 22 and recites instructions that enable the user to prepare a plurality of the diagrams corresponding to separate parts of an overall application and to specify relationships between individual ones of the diagrams. The Office asserts that this is found in column 8, lines 14-18 and column 9, line 53-57 of Bailey. However, these sections of Bailey disclose no such thing.

The sections of Bailey referred to by the Office discuss a single diagram. The Office appears to be considering the individual symbols in the diagram to be individual diagrams. However, this is inappropriate in the context of claim 27 because it clearly recites a plurality of different diagrams, not a single diagram, as well as a plurality of symbols. Thus, it would be inappropriate to read this claim language as encompassing the multiple symbols in a single diagram of Bailey.

Thus, claim 27 even further distinguishes over the prior art of record.

Turning to the rejections under 35 U.S.C. §103, Applicant respectfully traverses. The rejections of claims 24, 28, 29, 31, and 32 are based on a combination of Bailey and Milakovich and the rejection of claims 1-21, 23, and 25 are based on a proposed combination of Bailey, Milakovich, and Visio 2000.

The Office recites a large number of proposed combinations of features from Milakovich into the software of Bailey in connection with these claims. There is a fundamental problem in the proposed combination of Bailey and Milakovich. Particularly, there is no motivation in the prior art to make the proposed combination of Bailey and Milakovich. In addition, there are more

specific problems with respect to the alleged teachings of Bailey as applied to individual claim recitations.

For instance, claim 24 depends from claim 22 and adds fifth instructions that enable the user to denote one and only one object in the work space as a main object. The Office asserts that this is found in Milakovich at column 3, lines 39-41 and column 16, lines 36-39 and that it would have been obvious to combine these teachings with Bailey because Milakovich's teachings of denoting one object as a main object would help the user to easily understand the logical flow of execution control of the application program that is being developed.

There are two problems with this analysis. First, there is nothing in the prior art to suggest such a combination. Milakovich discloses a system and method for translating visual display object files from non-Component Object Model objects (hereinafter non-COM objects) to COM objects for plant control system software. Milakovich is not a system or method for defining or developing object oriented programming or any type of programming. Rather, it is a system for translating one type of programming object into another type of programming object. Particularly, as discussed in column 2, lines 48-63, the problem that Milakovich is addressing is the desire to convert visual display object files developed my means of one object-oriented programming language into the Microsoft Foundation Class (MFC) component object model (COM) specification. More particularly, Milakovich believes that companies that had software developed prior to the advent of the COM specification and have become accustomed to certain visual displays for running their plant control systems will

be reluctant to improve or replace that software with better software built on the COM model unless it maintains the same look and feel as the company's old software. Accordingly, Milakovich aims to develop a system and method for converting non-COM visual display object files to COM object to facilitate such conversions. Milakovich has nothing to do with modeling or developing software applications. The present invention (as well as Bailey) have nothing to do with converting visual objects in one format to visual objects in another software programming language format.

More specifically, there is no reason that one interested in modeling software for defining an application program so that a programmer can program it as in the present invention and Bailey would look to a software application for converting non-COM objects to COM objects for inspiration or, even if they did, find any inspiration therein. The references are simply inapposite to each other.

Accordingly, all of the 103 rejections are improper because there would be no motivation to make the proposed combination of Bailey and Milakovich that is at the core of each of these rejections.

Even further, and in any event, Milakovich does not teach that for which it has been cited. Particularly, referring to claim 24, contrary to the Office's assertion, column 3, lines 39-41 and column 16, lines 36-39 of Milakovich do not teach denoting one and only one object in the work space as a main object. Rather, column 3, lines 39-41 discloses:

> The reading circuit creates an intermediate file comprising: (1) the main picture containing a header record, (2) a main subpicture containing the plurality of related non-COM objects and (3) a data structure associated with data variables of the source visual display file.

However, the preceding paragraph recites:

> The reading circuit treats the source visual display file as a subpicture, the system thereby being a recursive translator. In the manner to be illustrated more fully, the source visual display file can be recursive. In such circumstances, it is best to treat the source visual display file as a subpicture (a picture within a picture).

Column 16, lines 36-39 do not add anything of significance relative to column 3.

There are several flaws with the Office's analysis. First, the claim language recites that the user is enabled to select one object, whereas Milakovich discloses a completely different paradigm in which the reading circuit automatically creates a file having a main picture and a subpicture.

Thus, there is no user selection in Milakovich, whereas user selection is the core of the present invention. Remember that the whole purpose of the present invention is to enable a user to design OOP programs. Secondly, in accordance with claims 22 and 24, the one and only one object that can be designated by a user is a graphical representation of object oriented programming logic that the user has dragged and dropped into a work space. In Milakovich, on the other hand, the "main picture" that the Office relies on essentially is the workspace. That is, it is the intermediate file being discussed and comprises essentially nothing but a header. Column 3, lines 39-41. The

source visual display that is being converted from non-COM to COM object is the "subpicture". Column 3, lines 39-41.

Hence, the analogy that the Office is attempting to make between Milakovich, on the one hand, and the present invention and Bailey, on the other hand, does not work.

Trying to cram the teachings of Milakovich into the present invention or Bailey is like trying to fit a square peg into a round hole. Milakovich simply deals with a very different task in a very different way than Bailey or the present invention.

This brings the argument full circle back to the fact that there simply is no reason to believe that a person of skill in the related arts could find it obvious to make the proposed combination in any event, even if Milakovich taught what the Office asserts. Particularly, a teaching of creating a file as a picture and inserting into it as a subpicture a display object that is to be converted between two different programming paradigms, quite simply, contains no teaching whatsoever that would be relevant to the task of determining how to effectively represent object oriented programming for purposes of enabling a software architect to graphically render an OOP application program. The Office is simply cutting and pasting a piece of Milakovich into Bailey purely based on hindsight reconstruction.

With respect to claim 28, which adds instructions that enable the user to include references associated with symbols in one diagram that identify at least one other diagram within which the object represented by that symbol also

appears. The Office cites the exact same portions of columns 3 and 16 of Milakovich as teaching this recitation.

In other words, when an object appears in two or more diagrams, the object can cross reference the two or more diagrams. These sections of Milakovich do not have anything to do with this topic. They do not discuss multiple diagrams, let alone multiple diagrams containing the same object. These sections of Milakovich are entirely irrelevant.

Claim 29 specifies that the instructions recited in claim 28 enable the user to specify in a first one of the diagrams the nature of the relationship of the representation of the object in the first diagram relative to the representation of the object in the second diagram, wherein the relationship between the object as represented in the first and second diagrams is selected from the group comprising: (1) the second diagram discloses additional details about the object in the first diagram; (2) the second diagram shows the object in a more abstract context than the first diagram; and (3) the object is the main object of the second diagram.

Again, the Office cites the exact same lines from columns 3 and 16 as disclosing these features. However, Applicant has reviewed these lines as well as the surrounding text and the entirety of Milakovich and finds nothing that even remotely resembles what is recited in claims 27 and 29. These sections of columns 3 and 16 of Milakovich teach that the reading circuit creates an intermediate file in the form of a main picture and places the source visual

display that is to have its non-COM objects converted into it as a subpicture. This is simply irrelevant to what is claimed in claims 27 and 29.

Furthermore, the Office seems to be of the impression that Milakovich is relevant to the present invention because the software disclosed by Milakovich deals with converting visual display objects between programming paradigms. However, while what Milakovich ostensibly pertains to graphically representing programming logic, it does so in a way that is utterly irrelevant to the present invention and Bailey.

Particularly, the present invention relates to a method of graphically representing object oriented programming logic. While the specification discloses an exemplary use of the invention to generate a GUI, this is merely exemplary. The object oriented programming logic that is developed can be anything, e.g., it could be a program that adds two numbers together. The graphical aspect of the present invention is the process of converting what in the program architect's head into a graphical representation of OOP. Milakovich, on the other hand, is a program for converting a graphical representation from one programming paradigm to another programming paradigm. Thus, in the present invention, (1) the process starts with the idea in the architect's head, and (2) uses the present invention (which can be a GUI in the software embodiment or a method using pencil and paper in a non-software embodiment) (3) to generate a graphical representation of programming logic. The present invention is all about step 2 and particularly about graphically representing OOP.

Milakovich, on the other hand, is, in many respects, the exact opposite. Particularly, in Milakovich, (1) the process starts with a GUI, (2) runs the data of the GUI through programming logic with little or no human intervention, let alone through pencil and paper or GUI-type interaction, and (3) outputs the same GUI it started with, but in a different programming paradigm. Milakovich is all about step 2, i.e., the process of making the conversion. However, note that step 2 essentially does not involve the use of graphical representations, which is the entire crux of the present invention..

Stated in summary, the present invention concerns the use of graphical representations in the process of generating programming logic. Milakovich, on the other hand, concerns converting a first graphical representation into another graphical representation. However, the processing between those two things that is the subject of Milakovich and that the Office is relying on does not significantly involve graphical representations. It basically is just programming logic running inside a computer. The fact that the input to this programming logic is a GUI and the output is a GUI is irrelevant. As noted above, in the present invention, the fact that the programming logic that is being developed in the example described in the specification happens to be a GUI is merely coincidence. It could be any OOP. Milakovich and the present invention essentially pertain to unrelated subject matter.

Looking at the obvious-to-combine issue in even simpler terms, Milakovich has nothing to do with developing object oriented programming logic. It simply takes a GUI in a first programming paradigm and converts it to a different

programming paradigm. Milakovich has nothing to do with graphically representing object oriented programming as in the present invention and Bailey. Milakovich has to do with graphically representing a graphical representation (in a different programming paradigm).

Accordingly, all of the obviousness rejections that rely on some combination of Bailey and Milakovich, namely, claims 1-21, 23-25, 28, 29, 31, and 32, are overcome because the proposed combination of Bailey and Milakovich categorically is not within the skill of a person of ordinary skill in the art. The two references are essentially inapposite.

Claim 32 recites that seventh instructions that restrict the user to using the sixth, eighth, and ninth symbols to connect to other object symbols. The Office asserted that this is taught in column 4, lines 35-40 of Bailey. However, this portion of Bailey discloses:

> Preferably, the developer connects one end of the wire construct to a terminal of a first control object symbol (referred to as the "source" control) and the second end to the terminal of a second control object symbol (referred to as the "sync" control).

This portion of Bailey simply teaches connecting two objects together with a third object. That is already recited in claim 22. Claim 32 recites entirely different instructions about restricting the use of certain symbols to certain tasks. This portion of Bailey does not have anything to do with that topic.

With respect to claim 31, which recites what the various symbols represent, the Office has asserted that Bailey discloses some of these symbols and that it would have been obvious to use the other symbols in view of Bailey.

While this conclusion is highly questionable, at a minimum, claim 31 is patentable because it depends from claims 22 and 24 and distinguishes over the prior art for at least all of the reasons set forth above in connection with those two claims.

The Office also rejected claims 1-21, 23, and 25 as unpatentable over Bailey in view of Milakovich and further in view of Visio 2000.

With respect to independent claim 1, the Office is essentially relying on the same teachings in Bailey and Milakovich and the same proposed combination as discussed above in connection with claim 22. Thus, claim 1 distinguishes over the prior art for largely the same reasons discussed above in connection with claims 22, 24, 28, 29, 31, and 32. Particularly, for instance, with respect to step 2 of claim 1 (selecting an object as a main object of the logic to be represented in the diagram) and step 3 of claim 1 (labeling the symbol with a label descriptive of the object's features), the Office is relying on combining Milakovich with Bailey. However, as described above in connection with claims 24 and 25, Milakovich neither teaches this nor is it obvious to make the proposed combination, in any event.

Accordingly, claim 1 patentably distinguishes over the cited prior art.

Claims 2-21 depend from claim 1 and therefore also distinguish over the prior art.

Likewise, claims 23 and 25, which depend from claim 22, distinguish over the prior art of record for at least all of the reasons set forth above in connection with claim 22. The additional citation of the Visio 2000 reference does not add

the teachings missing from the Bailey and Milakovich references discussed above in connection with claim 22.

Furthermore, at least some of the dependent claims add additional patentably distinguishing features. For instance, both claims 3 and 23 add denoting the main object in the diagram by drawing another symbol around the symbol for the main object. The Office asserts that this is disclosed in the prior art because one can do this with Visio 2000. Particularly, the Office asserts that Milakovich teaches selecting an object as the main object of the logic, but fails to teach drawing a symbol around it, but asserts that Visio 2000 teaches drawing a symbol corresponding to an object.

Applicant respectfully traverses. First, Milakovich does not teach selecting an object as a main object, as discussed above in detail in connection with claim 22. Furthermore, the fact that Visio 2000 teaches drawing a symbol corresponding to an object is utterly irrelevant. The claim does not recite drawing a symbol corresponding to an object, but drawing a very specific symbol corresponding to a very specific object.

As Applicant noted in response to the previous rejection, which relied solely on Visio 2000 for rejecting the claims, Visio 2000 may disclose software that can be used to practice the present invention, but this is a very different thing from disclosing the present invention. To repeat the analogy mentioned in Applicant's previous response, this rejection is like saying that disclosing a hammer and a chisel is equivalent to disclosing Michelangelo's David.

Claim 4 recites that step 7 comprises drawing a circle completely around the main object symbol. Again, the Office simply cites Visio 2000 asserting that one can draw a circle around an object using Visio 2000. However, as described above, at best, Visio 2000 merely discloses a program that would allow one to practice the step, but it clearly does not suggest the step.

In fact, without belaboring the point, the Office's rejections of claims 4, 5, 7, and 8 suffer from the same faulty reasoning with respect to Visio 2000.

With respect to claim 9, which recites repeating steps (1)-(5) to prepare a plurality of separate diagrams corresponding to separate parts of an overall application, the Office asserts that this is found in column 8, lines 14-18 of Bailey. However, lines 14-18 of column 8 of Bailey describe making one diagram comprising a plurality of symbols. This is completely different. Claim 9 further recites that a first object is the main object appearing in at least a first one of the diagrams and is not a main object appearing in at least a second one of the diagrams. The Office asserts that Milakovich teaches this at the same column 3, lines 38-41 so heavily relied on with respect to claim 22. However, as described above, Milakovich does not even discuss multiple diagrams. In fact, Milakovich does not discuss diagrams at all. Rather, Milakovich describes a file.

Most of the remaining claims recite even further distinguishing features. However, in view of the large number of distinguishing features already discussed above in connection with the claims from which the remaining dependent claims depend, Applicant will not belabor the point by going through each distinction of each of the many claims.

Finally, the Office rejected claims 30, 33, and 34 under 35 U.S.C. §103(a) as being obvious over Bailey alone. These claims all depend, either directly or indirectly from claim 22. Therefore, they distinguish over Bailey for at least all of the reasons set forth above in connection with claim 22.

**Conclusion**

In view of the foregoing remarks, this application is now in condition for allowance. Applicant respectfully requests the Office to issue a Notice of Allowance at the earliest possible date. The Examiner is invited to contact Applicant's undersigned counsel by telephone call in order to further the prosecution of this case in any way.

Respectfully submitted,

Dated: January 29, 2008

*/Theodore Naccarella/*

Theodore Naccarella, Reg. No. 33,023
Synnestvedt & Lechner LLP
1101 Market Street; Suite 2600
Philadelphia, PA 19107-2950
Telephone: (215) 923-4466
Facsimile: (215) 923-2189
Attorneys for Applicant

TXN:pmf